# Lab: Advanced Collections

Problems for exercises and homework for the [“Programming Fundamentals Extended” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

Check your solutions here: <https://judge.softuni.bg/Contests/429>.

# Multi-Dictionaries, Nested Dictionaries

## Average Student Grades

Write a program, which reads the **name** of a student and their **grades** and **adds** them to the **student record**, then **prints** **grades** along with their **average grade**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 7  Ivancho 5.20  Mariika 5.50  Ivancho 3.20  Mariika 2.50  Stamat 2.00  Mariika 3.46  Stamat 3.00 | Ivancho -> 5.20 3.20 (avg: 4.20)  Mariika -> 5.50 2.50 3.46 (avg: 3.82)  Stamat -> 2.00 3.00 (avg: 2.50) |
| 4  Vladimir 4.50  Petko 3.00  Vladimir 5.00  Petko 3.66  Ivan 4.33 | Vladimir -> 4.50 5.00 (avg: 4.75)  Petko -> 3.00 3.66 (avg: 3.33) |
| 5  Gosho 6.00  Gosho 5.50  Gosho 6.00  Ivan 4.40  Petko 3.30  Petko 4.50 | Gosho -> 6.00 5.50 6.00 (avg: 5.83)  Ivan -> 4.40 (avg: 4.40)  Petko -> 3.30 (avg: 3.30) |

### Hints

* Use a **dictionary** (string 🡪 list of floats)
* Check if the name exists before adding the grade. If it doesn’t, add it to the dictionary.
* Pass through all **key-value pairs** in the dictionary and print the results. You can use average() method to quickly calculate the average value from a list.

## Cities by Continent and Country

Write a program to read **continents**, **countries** and their **cities**, put them in a **nested dictionary** and **print** them.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 9  Europe Bulgaria Sofia  Asia China Beijing  Asia Japan Tokyo  Europe Poland Warsaw  Europe Germany Berlin  Europe Poland Poznan  Europe Bulgaria Plovdiv  Africa Nigeria Abuja  Asia China Shanghai | Europe:  Bulgaria -> Sofia, Plovdiv  Poland -> Warsaw, Poznan  Germany -> Berlin  Asia:  China -> Beijing, Shanghai  Japan -> Tokyo  Africa:  Nigeria -> Abuja |
| 3  Europe Germany Berlin  Europe Bulgaria Varna  Africa Egypt Cairo | Europe:  Germany -> Berlin  Bulgaria -> Varna  Africa:  Egypt -> Cairo |
| 8  Africa Somalia Mogadishu  Asia India Mumbai  Asia India Delhi  Europe France Paris  Asia India Nagpur  Europe Germany Hamburg  Europe Poland Gdansk  Europe Germany Danzig | Africa:  Somalia -> Mogadishu  Asia:  India -> Mumbai, Delhi, Nagpur  Europe:  France -> Paris  Germany -> Hamburg, Danzig  Poland -> Gdansk |

### Hints

* Use a **nested** **dictionary** (string 🡪 (dictionary 🡪 list of strings))
* Check if the continent exists before adding the country. If it doesn’t, add it to the dictionary.
* Check if the country exists, before adding the city. If it doesn’t, add it to the dictionary.
* Pass through all **key-value pairs** in the dictionary and the values’ key-value pairs and print the results.

# Sets

## Record Unique Names

Write a program, which will take a list of **names** and print **only** the **unique** names in the list.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 8  Ivan  Pesho  Ivan  Stamat  Pesho  Alice  Peter  Pesho | Ivan  Pesho  Stamat  Alice  Peter |  | 7  Lyle  Bruce  Alice  Easton  Shawn  Alice  Shawn  Peter | Lyle  Bruce  Alice  Easton  Shawn |  | 6  Roki  Roki  Roki Roki  Roki  Roki | Roki |

### Hints

You can store the names in a set of string to extract only the unique ones.

## Group Continents, Countries and Cities

Write a program, which receives **continents**, **countries** and **city names** and **prints them**, grouped by continent, country and city, all in **alphabetical order**. No duplicates are allowed.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 9  Europe Bulgaria Sofia  Asia China Beijing  Europe Bulgaria Sofia  Europe Poland Warsaw  Europe Germany Berlin  Europe Poland Poznan  Europe Poland Warsaw  Asia China Beijing  Asia China Shanghai | Asia:  China -> Beijing, Shanghai  Europe:  Bulgaria -> Sofia  Germany -> Berlin  Poland -> Poznan, Warsaw |
| 7  Europe Bulgaria Sofia  Europe Bulgaria Sofia  Europe Bulgaria Plovdiv  Europe Bulgaria Sofia  Asia China Beijing  Asia India Mumbai  Europe Turkey Istanbul | Asia:  China -> Beijing  India -> Mumbai  Europe:  Bulgaria -> Plovdiv, Sofia  Turkey -> Istanbul |
| 10  Europe Turkey Ankara  Asia China Shenyang  Asia India Delhi  Asia India Pune  Asia Japan Yokohama  Asia China Suzhou  Europe Russia Moscow  Europe Turkey Ankara  Asia Japan Tokyo  Asia India Pune | Asia:  China -> Shenyang, Suzhou  India -> Delhi, Pune  Japan -> Tokyo, Yokohama  Europe:  Russia -> Moscow  Turkey -> Ankara |

### Hints

* Since everything needs to be sorted and unique, you can use a Sorted Dictionary (string -> Sorted Dictionary (string -> Sorted Set (string))). That way, since we’re using **dictionaries**, we **won’t have** any **duplicated** **continents** or **countries**.
* By using a Sorted Set to store the city names, we ensure we **won’t have** any **repeating** cities. Since everything is sorted, the alphabetical order is taken care of.